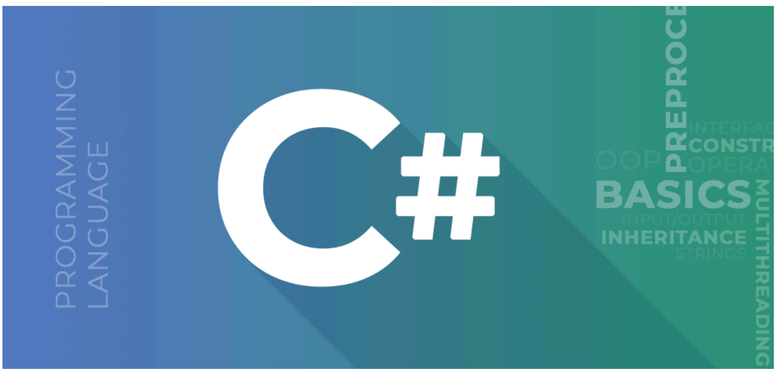
RazorSlices - 具有 ASP.NET 核心最小 API 的 Razor 视图

作者：微信公众号：【架构师老卢】 1-30 10:23 10



概述：随着 .NET 8 的推出，Minimal API 继续扩展其产品/服务，慢慢地提供与之前框架类似的功能集，包括 ASP.NET Core MVC 和 Razor Pages。当社区中有人（嗨，Johnathan Channon）向我指出一个引人入胜的 RazorSlices 项目时，我感到很惊讶。RazorSlices 由 Damian Edwards 编写，他是 ASP.NET Core 团队成员，具有使用 Razor 视图引擎的历史。他的目标是将 Razor 功能带给最小 API 采用者，其方式与 ASP.NET Core MVC、Razor Pages 和 Blazor 用户目前所

随着 .NET 8 的推出，Minimal API 继续扩展其产品/服务，慢慢地提供与之前框架类似的功能集，包括 ASP.NET Core MVC 和 Razor Pages。当社区中有人（嗨，Johnathan Channon）向我指出一个引人入胜的 RazorSlices 项目时，我感到很惊讶。RazorSlices 由 Damian Edwards 编写，他是 ASP.NET Core 团队成员，具有使用 Razor 视图引擎的历史。他的目标是将 Razor 功能带给最小 API 采用者，其方式与 ASP.NET Core MVC、Razor Pages 和 Blazor 用户目前所享受的方式类似。

在这篇文章中，我们将了解如何在 ASP.NET Core 应用程序中设置 RazorSlices，并考虑其使用的利弊。

RazorSlices 入门

在最小 API 项目中，首先安装 RazorSlices 包。为此，可以使用该命令或在文件中添加相应的命令。dotnet add package RazorSlicesPackageReference.csproj

```
<PackageReference Include="RazorSlices" Version="0.7.0" />
```

接下来，在 ASP.NET Core 应用程序的根目录下创建一个文件夹。它可以被命名为任何你喜欢的名字。我选择了文档推荐的内容。在此目录中，你将放置要在最小 API 终结点中使用的所有 Razor 视图。Slices

在该文件夹中，创建一个名为 的新文件。视图导入文件允许您为文件夹中的所有视图（包括命名空间、基类和标记帮助程序）创建共享上下文。这是我文件的内容。Slices_ViewImports.cshtml

```
@inherits RazorSliceHttpResult

@using System.Globalization;
@using Microsoft.AspNetCore.Razor;
@using Microsoft.AspNetCore.Http.HttpResults;

@tagHelperPrefix __disable_tagHelpers__:
@removeTagHelper *, Microsoft.AspNetCore.Mvc.Razor
```

在查看此文件时，您会注意到两件奇怪的事情：

1. 该值是使用我的最小 API 使用的类型。@inheritsRazorSliceHttpResultIResult
2. 标记帮助程序将被删除并进行模糊处理。这是因为此模型不支持它们。仅期望通过进行一些修改或让步将 Razor Pages 实现移植到 RazorSlices。

接下来，您需要在文件夹中创建一个新视图。Hello.cshtmlSlices

```
@inherits RazorSliceHttpResult<DateTime>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Hello from Razor Slices!</title>
</head>
<body>
<p>
    Hello from Razor Slices! The time is @Model
</p>
</body>
</html>
```

请注意，视图也使用该属性，但这次使用泛型类型定义。泛型类型参数是实例的类型。@inherit@Model

最后，您需要在最小 API 终端节点中使用该视图。

```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/", () =>
    Results.Extensions.RazorSlice("/Slices/Hello.cshtml", DateTime.Now));

app.Run();
```

就是这样！现在，你正在通过最小 API 终结点呈现 Razor 视图。

对于使用 JetBrains 工具的用户，请多花一点时间安装 **JetBrains.Annotations。**

```
dotnet add package JetBrains.Annotations
```

然后，修改代码，如下所示。

```
using JetBrains.Annotations;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/", () =>
    Slice("/Slices/Hello.cshtml", DateTime.Now));

app.Run();

IResult Slice<T>([AspMvcView]string viewPath, T model)
    => Results.Extensions.RazorSlice(viewPath, model);
```

这样做的好处是，您现在可以直接从 C# 代码 + 单击进入您的视图。这应该有助于您的工作效率。之所以有效，是因为您需要通过绝对路径引用 RazorSlice 视图，而 JetBrains Rider 和 ReSharper 可以理解这种行为。真是太棒了，对吧？Ctrl

RazorSlices 的优点

此处最明显的好处是，现在可以使用 Razor 在最小 API 终结点中生成 HTML 结果。我可能有偏见，但我喜欢 Razor 语法及其对 Web 开发体验的好处。

另一个显著的好处是 Razor 视图也可以使用该属性，使你能够访问 HTML 呈现过程中的依赖项。@inject

如果您关心性能问题，可以使用 ，它可以帮助降低与查找视图相关的成本。对于希望获得更多性能的人来说，这可能是一个轻松的胜利。说到性能，有计划添加提前（AOT） 支持。SliceFactory

RazorSlices的缺点

让我们谈谈使用 RazorSlices 的缺点，因为有一些缺点。

如前所述，如果您正在寻找 MVC 或 Razor Pages 的一对一替代品，您会感到失望。它意味着不是任何一种方法的直接替代品。这在使用 TagHelpers 或 View Components 等功能时很明显。

该库目前不支持部分视图，因此无法进行视图组合和组件化。它可能会进入包中，但目前不可用。这包括与 MVC 关联的视图基础结构，如布局和布局部分。正如我之前所说，这不是一个直接的替代品。

新的范式意味着抛弃过去的范式。如果依赖于第三方 Razor 库，则它们可能无法在此当前迭代中工作。

[结论]

如果你正在寻找一种方法来在 API 终结点中将 HTML 呈现为比字符串插值高出一步的方法，那么 RazorSlices 值得一试。它提供了你喜欢的 Razor 语法的基础知识，包括表达式、控制结构、循环和代码块。对于大多数人来说，这足以实现任何目标。也就是说，如果你正在寻找一个 ASP.NET Core MVC的替代品，这个库远非如此，但没关系。